WEB SCRAPING AMB PYTHON

Una anàlisi textual de la premsa escrita

Grau d'Estadística Aplicada - UAB

Joan Gasull Jolis

Tutor: Albert Ruiz Cirera

CONTINGUT

RESUM

L'objectiu fonamental d'aquest treball és crear una eina que, donada una notícia, predigui a quin diari pertany. S'ha utilitzat "web scraping" per a obtenir una base de dades de notícies de forma massiva i el més automàtica possible. El classificador de notícies utilitza models basats en el Teorema de Bayes, i en el treball s'estudia com maximitzar-ne el potencial predictiu. S'arriba a la conclusió que, en els casos estudiats i només considerant el vocabulari utilitzat, aquesta tècnica de classificació dóna uns resultats molt precisos.

INTRODUCCIÓ

Resumint breument el projecte, podem entendre'l com una descarrega automàtica de notícies de diferents diaris, amb l'objectiu de crear una eina per a predir i classificar a quin diari pertanyen les notícies segons el vocabulari que utilitzen.

Per a tal efecte, s'ha emprat el Python com a llenguatge de programació, i dues biblioteques principals: Beautiful Soup, orientada a l'obtenció de les dades de la web i el seu posterior processament, i Scikit-Learn, que inclou les eines per a construir el classificador utilitzat per a predir a quin diari corresponen les notícies. Per a avaluar el classificador s'ha utilitzat la validació externa, és a dir, un conjunt de notícies reservat per a comprovar l'eficiència d'aquest, i que no s'han utilitzat per ajustar el model.

Els diaris seleccionats han estat alguns dels principals mitjans de premsa escrita d'Espanya: El Mundo, La Razón, ABC, El Diario, Público i El País, i s'han obtingut gairebé 1000 notícies per a cadascun, corresponents als darrers anys i de les seccions de Política, Internacional i Nacional, per a tal de no introduir un biaix relacionat amb la temàtica o el temps.

Aquest informe està basat en un projecte estadístic principalment orientat a la programació, així que la major part de l'esforç i temps invertits en ell es troben als annexos, on es presenta el codi que ha generat tots els resultats que aquí apareixen.

Objectius del Treball

A continuació s'enumeren algunes dels principals fites que es busca assolir amb aquest projecte, així com alguns dels interrogants plantejats a l'inici de la recerca:

- Construir una eina que permeti classificar, mitjançant exclusivament el text, a quin diari pertanyen les notícies.
- Maximitzar el potencial i l'efectivitat del classificador.
- Estudiar quins són els paràmetres que, individualment i conjunta, ajuden a maximitzar l'efectivitat a l'hora de fer prediccions.
- Estudiar teòricament les principals eines que s'utilitzen per a l'anàlisi del text, i explorar-ne els seus avantatges i inconvenients.
- Aprendre un nou llenguatge de programació.
- Realitzar un projecte estadístic complet a gran escala, des del disseny i la recollida de dades fins a l'anàlisi i l'extracció de resultats.

Motivacions

Les motivacions principals per a dur a terme aquest projecte són fonamentalment dues. En primer lloc, aprendre un nou llenguatge de programació des de zero, i les seves aplicacions orientades a l'anàlisi de dades, l'estadística, i més àmpliament a la ciència de dades. El projecte podria haver-se desenvolupat igualment en R sense complicacions afegides, i en una quantitat de temps més reduïda donada l'experiència que s'obté al grau d'estadística en aquest llenguatge, però considerant l'oportunitat, es va decidir aprofitar el treball de fi grau per a impulsar l'aprenentatge d'aquest nou llenguatge. En segon lloc, l'exploració de tècniques que no apareixen directament al grau, però que estan estretament vinculades a al contingut d'aquest: El "Web scraping", en tant que l'obtenció automàtica de dades de fonts on-line permet explotar i analitzar dades altrament inaccessibles, i els classificadors basats en Naive Bayes, una eina conceptualment molt senzilla però amb uns resultats molt potents, que permet introduir-se a l'anàlisi de texts sense la necessitat de coneixements específics.

En conjunt, les tres dimensions (Python, "Web scraping" i eines de classificació) són competències molt valorades en el sector de l'estadística professional, i que van més enllà del contingut de l'estadística més clàssica que s'imparteix al grau. Així doncs, aprofitant el projecte de final de grau, s'ha buscat un primer contacte amb aquestes noves eines i obtenir un complement a la formació ja adquirida.

Agraïments

Estic especialment agraït a l'Albert Ruiz Cirera, el meu tutor, sobretot per deixar-me total llibertat a l'hora de plantejar el treball i deixar-me explorar àmbits que s'allunyen de la perspectiva més acadèmica de l'estadística. També a parella, pares i amics per aguantar interminables discursos sobre la potència del classificador, el nombre de scraping que havia aconseguit o els meus intents per explicar el funcionament d'un Spider.

WEB SCRAPING

Com s'ha assenyalat abans, aquesta és la primera fase del treball, que consisteix, en primer lloc, en l'obtenció de les dades amb les quals s'ha construït el classificador, i en segon lloc, en la neteja i adaptació de les dades per a una anàlisi correcta.

El "Web scraping" és, en poques paraules, una tècnica d'extracció d'informació de llocs web. Per a tal efecte, se simula la navegació d'un humà pel lloc web, i se selecciona i emmagatzema mitjançant un programa la informació que es considera rellevant per a l'estudi. L'ús més popular d'aquesta tècnica és l'extracció del contingut de webs generades dinàmicament a través d'una base de dades, sobretot en el referent a l'estudi de preus.

Biblioteca Beautiful Soup

EXTRACCIÓ DEL TEXT D'UNA NOTÍCIA

Beautiful Soup[1] és una biblioteca de Python orientada a obtenir el codi HTML o XML de pàgines web i a navegar dins d'elles. Aconseguir el codi és immediat a partir de l'enllaç a la web, i la biblioteca permet recórrer l'estructura en arbre del codi HTML saltant entre branques o explorant-les en profunditat, sempre de manera ordenada en funció de les "etiquetes" de cada secció. Aquestes funcionalitats, molt útils per a aprendre a utilitzar la biblioteca i obtenir informació massivament de pàgines web, sempre que aquestes estiguin estructurades de la mateixa manera, no són l'eina més adequada per a obtenir el text de les notícies. En canvi, sí ho és l'obtenció de tot el contingut present en determinades etiquetes, sense necessitar informació sobre l'estructura de l'arbre. Així doncs, prèvia exploració del codi HTML de cada diari, se seleccionen totes les etiquetes dins les quals es trobi el text de la notícia, i mitjançant un bucle s'emmagatzema en una única cadena de caràcters, que contindrà tot el codi HTML de la secció on es troba la informació rellevant.

Finalment es va optar per netejar el codi HTML seccionant la cadena de caràcters en la seva totalitat com si es tractés d'un *string* normal, és a dir, rebutjant tota la informació que es trobés continguda dins de "<>", que forma part del codi HTML. Això implica deixar d'utilitzar la biblioteca i els seves funcions per a tal efecte, però resulta extremadament efectiu en pràcticament totes les ocasions, i prenent precaucions adequades, permet obtenir el cos de la notícia independentment del diari del qual provingui, és a dir, sense considerar l'estructura específica de cada web.

A continuació es plantegen les dues situacions que es poden trobar a l'hora de buscar automàticament notícies en una pàgina web: els llistats de notícies o la necessitat d'un Spider.

LLISTAT DE NOTÍCIES (PÚBLICO, EL DIARIO I ABC)

Certs diaris ofereixen a l'usuari una direcció web on trobar, llistades per data de publicació, totes les notícies d'una certa temàtica. Aquesta estructura facilita enormement la tasca d'obtenir els enllaços, ja que, mitjançant la mateixa direcció URL i iterar una variable, podem accedir a totes les pàgines. Per exemple, l'enllaç per accedir a les notícies de política del Público és el següent:

http://www.publico.es/page/ultimas-noticias-politica.html?page=0

Llegint aquest enllaç com una cadena de caràcters es pot anar substituint el zero final pel nombre desitjat, i posteriorment anar emmagatzemant totes les notícies de cadascuna de les pàgines.

[1] Pot consultar-se la documentació oficial del paquet a https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Aquest mètode, a part de ser més clar i senzill de programar, permet garantir que totes les notícies corresponguin a un període determinat, ja que els llistats estan ordenats de més recent a més antic, el qual pot reduir el biaix associat a que les notícies obtingudes per a diferents diaris fan referència a un període temporal diferent.

SPIDER (EL PAÍS, LA RAZÓN I EL MUNDO)

Si la web de la qual es vol obtenir informació no té un llistat de notícies que es pugui recórrer fàcilment, es necessita el que col·loquialment es coneix com un Spider o "Web crawler", és a dir, un programa que et recorri una pàgina determinada buscant-te enllaços (en aquest cas enllaços a notícies) i els emmagatzemi, obri un d'ells i realitzi el mateix procediment, i així successivament, fins a obtenir el nombre d'enllaços desitjats. Per tal d'evitar processar pàgines per duplicat és interessant emmagatzemar tots aquells enllaços que ja s'han explorat, i per evitar entrar en pàgines on és poc probable trobar les notícies desitjades, filtrar quins enllaços són vàlids.

Aquest procediment és evidentment molt més lent que recórrer un llistat estructurat de notícies, ja que res garanteix que a les webs explorades hi hagi enllaços rellevants, i obtenir una quantitat elevada d'enllaços diferents no és immediat. A més a més, donat que el Spider obté tots els enllaços d'una determinada web, s'han de definir uns criteris de filtratge molt exhaustius per evitar que reculli enllaços que compleixin alguna de les següents característiques:

- Surti del domini de la web que estem explorant, per exemple, "*http://politica.elpais.com/*".
- Es tracti d'una notícia en un format no rellevant per a l'anàlisi, com vídeo-notícies, enquestes, contingut interactiu o d'altres formats.
- Remeti a una web o secció d'una notícia que ja s'hagi emmagatzemat anteriorment, per exemple, a la secció de comentaris d'una notícia o una ampliació del seu contingut.
- Sigui un enllaç trencat.

En definitiva, a banda d'implementar un codi que consideri aquestes excepcions, es necessita una gran quantitat de temps dedicat a l'assaig i error, per a tal de detectar totes aquelles irregularitats que poden impedir el funcionament correcte del programa o l'emmagatzematge d'una observació no pertinent per al classificador. Hi ha algunes d'aquestes característiques que són comunes entre diaris, i d'altres que s'han d'explorar un a un, ja sigui per la diferent estructura dels continguts de la web o l'existència de casos particulars.

BASE DE DADES

Per a la construcció de la base de dades definitiva de notícies, cal combinar dos dels processos ja anomenats. En primer lloc l'obtenció d'enllaços, ja sigui mitjançant un llistat de notícies estructurat o gràcies a la construcció d'un Spider, i en segon lloc l'extracció de la informació útil dels enllaços obtinguts.

Combinar el llistat de notícies i l'obtenció del text és un procés que es pot fer sense un esforç addicional en dues fases diferenciades (primer tots els enllaços i després l'obtenció de tots els texts a partir dels enllaços), però utilitzar un Spider genera una situació diferent. Malgrat l'estructura òptima en termes d'eficiència i càlcul de l'ordinador seria realitzar els dos processos simultàniament, és a dir, anar obtenint el text de les notícies rellevants a la vegada que s'exploren els enllaços mitjançant el Spider (ja que s'estalvia llegir els codis HTML per duplicat), fer-ho tot a la vegada complica la detecció de possibles errades en l'obtenció de les dades (especialment enllaços), i incrementa notablement el temps necessari per a resoldre les incidències, així que el procés es va realitzar seqüencialment en ambdós escenaris.

Així doncs, l'estructura general de la creació de la base de dades final és la següent[2] (on cada registre consta només de dos variables; el cos de la notícia i el diari al qual pertany):

1. Obtenir tots els enllaços (a través d'un Spider o un llistat de notícies).
2. Obtenir el text de cadascun dels enllaços.
3. Guardar tots els texts obtinguts en una base de dades.
4. Combinar les bases de dades de cadascun dels diaris en una.
5. Netejar la base de dades conjunta.
6. Guardar la base de dades definitiva.

Consideracions sobre les Dades

Un cop explicades les fases principals del procés, és convenient repassar alguns detalls i particularitats que s'han de considerar per tal de donar coherència a la mostra final o evitar errors en el programa, que per la seva puntualitat no escauen a d'altres apartats.

En primer lloc és necessari controlar i tenir present la codificació del text amb la qual es processen les dades. És realment important ja que si les notícies dels diferents diaris no estan codificades de la mateixa manera, el classificador interpretarà la mateixa paraula en ambdós diaris com si no ho fos, i lògicament inflarà erròniament la precisió. Per a tal d'evitar aquests problemes, s'ha treballat en tot moment amb la codificació UTF-8.

En segon lloc, s'ha realitzat la neteja del codi HTML mitjançant les funcions bàsiques del python i no les de la biblioteca Beautiful Soup. Així doncs, s'ha extret del codi tota aquella informació que es trobés entremig dels símbols "<" i ">" (i en alguns casos entre "{" i "}"), el qual és una manera senzilla i ràpida de quedar-se tan sols amb el text de les notícies. Afegint una excepció en cas que en el cos de la notícia s'hagués utilitzat algun d'aquests dos signes, el qual podria generar un error al no trobar-los aparellats.

En tercer lloc cal assenyalar que alguns dels processos realitzats de neteja de les dades (per exemple la eliminació d'espais en blanc i salts de pàgina, valors numèrics o de valors nuls) s'han dut a terme en múltiples ocasions, algunes vegades per duplicat, sobre un mateix conjunt de dades, donat que l'estructura en la qual s'ha construït el procés d'anàlisi de dades i el codi per a tal efecte no s'ha escrit de forma lineal, i per a validar algun dels processos intermedis eren necessàries aquestes accions.

En quart lloc, una de les consideracions que cal tenir en compte és el moment temporal en el qual es van escriure les notícies. Aquest factor s'ha intentat controlar limitant l'entrada de les notícies al Spider segons el seu any de publicació (en cap cas anterior al 2015), i en els llistats no s'han extret prou notícies per arribar més enllà d'aquest llindar. No controlar aquesta dimensió podria generar algun tipus de biaix associat als temes que són notícia en algun període i en un altre no ho són, o podria haver-se modificat l'estil d'escriptura d'un diari concret, el qual generaria massa variabilitat dins del mateix grup. No es pot garantir que per a la realització d'aquest treball es disposi d'una mostra perfectament equilibrada temporalment, ja sigui pel funcionament intrínsec d'un Spider o per la freqüència de publicació dels diferents diaris en diferents categories. Controlar millor aquest aspecte donaria més validesa al classificador i permetria descartar com a possible fonts d'incertesa el desajust temporal. En tot cas, s'ha decidit considerar el ventall temporal de 2-3 anys com a un marge acceptable d'error.

En últim lloc, s'han hagut d'eliminar algunes notícies puntualment a mà, que havien estat recollides més d'una vegada malgrat els filtres aplicats per a evitar-ho. Una exploració visual de les dades permet detectar les incidències més significatives d'aquest tipus. Si bé aquesta detecció no s'ha realitzat en molta

[2] Els passos 1, 2 i 3 es repeteixen per a cadascun dels diaris.

profunditat, l'aparició de notícies repetides un o pocs cops no sembla perjudicar en gran mesura els resultats globals del model, donat que s'assoleix un nivell força elevat de precisió sense haver realitzat aquestes comprovacions.

Mostra Final

La base de dades obtinguda després de tot el procés descrit és la següent:

Nom	Registres	Mitjana de paraules
El Diario	905	406.60
Público	900	721.32
ABC	863	521.41
La Razón	832	537.25
El Mundo	832	548.68
El País	752	900.40
Total	**5084**	**599.47**

Taula 1. Mostra Final

Entre tots els registres, la mostra consta de 3.047.750 paraules, i 18.408.198 caràcters.

CLASSIFICADORS

Classificador Naive Bayes

Els classificadors emprats en aquest treball es troben dins de la categoria dels classificadors Naive Bayes. Són mètodes d'aprenentatge supervisat, és a dir, que requereixen disposar de dades etiquetades en base a les quals generar mecanismes per a predir la seva pertinença a aquests grups, i que es fonamenten en el teorema de Bayes, i en l'assumpció (naïf) que cada parell de paraules és independent entre sí. Es considera naïf per la pròpia naturalesa del text, ja que la presència d'unes paraules sí condiciona la presència d'unes altres. Malgrat aquesta assumpció no sigui estrictament certa, els classificadors basats en aquesta idea donen uns bons resultats, fins i tot millors que d'altres models més sofisticats (Hand & Yu, 2001). Un dels exemples més clàssics d'aquests tipus de classificadors (Metsis et al., 2006) és el filtratge de correus electrònics segons si es tracta de spam o no. A continuació mostrem el procés[3] emprat, segons la documentació oficial de la biblioteca:

El teorema de Bayes ens diu que:

$$P(C_k \mid x_1, \dots, x_n) = \frac{P(C_k)P(x_1, \dots, x_n \mid C_k)}{P(x_1, \dots, x_n)}$$

Partint de la suposició d'independència (naïf) que:

$$P(x_i \mid x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n, C_k) = P(x_i \mid C_k)$$

Obtenim la següent expressió:

$$P(C_k \mid x_1, \dots, x_n) = \frac{P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)}{P(x_1, \dots, x_n)}$$

Donat que el denominador és constant en base a la mostra, trobem l'equació final on el primer terme és proporcional al segon:

$$P(C_k \mid x_1, \dots, x_n) \propto P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$$

Ja tenim doncs una eina de classificació, ja que la component C_k que maximitzi la probabilitat condicionada a tota la mostra, serà aquella que assignem a tot el conjunt $x_1, \dots, x_n$.

$$\widehat{C_k} = argmax(P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k))$$

A continuació mostrem els dos classificadors que s'han utilitzat per a realitzar aquest treball.

CLASSIFICADOR MULTINOMIAL

El classificador Multinomial basat en el supòsit naïf d'independència entre variables es construeix mitjançant vector de la forma $\theta_k = (\theta_{k_1}, \dots, \theta_{k_n})$ per a cada diari k, on n és el nombre total de paraules diferents, i θ_{k_i} representa la probabilitat $P(x_i \mid C_k)$.

Els paràmetres θ_{k_i} s'estimen segons una versió suavitzada (segons el paràmetre α) de la màxima versemblança, segons la freqüència relativa amb la qual apareixen a cada notícia:

[3] La notació emprada és la següent:
C_k correspon a cadascun dels diaris, on k = 1,2,3,4,5,6
x_i és una variable indicadora corresponent a cadascuna de les paraules diferents (o grups de paraules, segons l'ordre del n-grama utilitzat) en tota la mostra, fins a n.

$$\widehat{\theta_{k_i}} = \frac{N_{k_i} + \alpha}{N_k + \alpha n}$$

On:

$N_{k_i} = \sum_{x \in T} x_i$ és el nombre de vegades que la paraula i apareix al diari k en el conjunt d'entrenament.

$N_k = \sum_{i=1}^{T} N_{k_i}$ és el recompte total de totes les paraules al diari k.

L'expressió final, utilitzant els resultats obtinguts a l'apartat anterior, és la següent:

$$P(C_k \mid x_1, \dots, x_n) \propto P(C_k) \prod_{i=1}^{n} \frac{N_{k_i} + \alpha}{N_k + \alpha n}$$

El paràmetre de suavitzat α se situa en [0,1], i s'anomena de Laplace quan és igual a 1, i de Lidstone altrament. Malgrat que teòricament el 0 és un valor possible, genera errors de càlcul entrenar el classificador sense paràmetre de suavitzat, ja que quan un dels termes dóna zero, és a dir, quan una paraula no apareix a un diari concret, s'ha de calcular el logaritme de zero, el qual atura el programa[4]. Tot i així, com s'estudia en el següent apartat, els valors propers a zero són aquells que ofereixen uns millors resultats predictius.

CLASSIFICADOR DE BERNOUILLI

El classificador Naive Bayes de Bernouilli es basa en l'assumpció que les diferents paraules de cada notícia són variables binàries independents. Això es tradueix en la següent expressió per a cada paraula concreta, on x_i és binària segons si la paraula concreta apareix o no en la notícia:

$$P(x_i \mid C_k) = P(i \mid C_k)^{x_i}(1 - P(i \mid C_k))^{(1-x_i)}$$

Que aplicat a tota la mostra obtenim:

$$P(x_1, \dots, x_n \mid C_k) = \prod_{i=1}^{n} {p_{ki}}^{x_i}(1 - p_{ki})^{(1-x_i)}$$

Així doncs, en el classificador de Bernouilli, en contraposició al classificador Multinomial, que es fonamenta en les freqüències dels termes en cada notícia, penalitza la no aparició d'una certa paraula a una notícia, és a dir, també es dóna un pes a tots aquells termes que no ocorren en una notícia, però que han aparegut en d'altres.

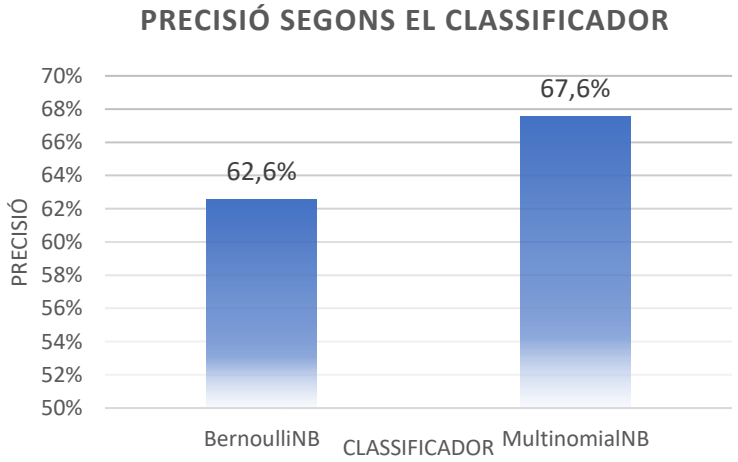
[4] Una descripció més detallada de l'algoritme pot trobar-se a: https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

Paràmetres

A continuació es mostren els resultats mitjans de classificació per a cadascun dels paràmetres estudiats, analitzats marginalment. El procés de simulació es basa en entrenar 18.000 classificadors diferents[5], procés que ha consumit més de 90 hores de càlcul[6]. Cada combinació possible de paràmetres o classificadors (N-Grames, mínim suport, tipus de classificador, suavitzat i mida mostral) s'ha repetit 15 cops per a minimitzar l'impacte de la selecció de la mostra training/test, i obtenir un conjunt d'estadístics referents a aquesta (Mitjana de predicció, mediana, variància, etc.). Per a mostrar gràficament els resultats s'ha seccionat l'eix vertical al 50 %, ja que es busca emfatitzar les diferències entre els diferents paràmetres. També cal considerar que l'ordenació de l'eix x no correspon a l'ordre natural de les variables, sinó a quin valor mitjà de predicció ofereixen, i estan ordenades amb aquest criteri.

TIPUS DE CLASSIFICADOR

En primer lloc es mostra la diferència en potencial predictiu dels dos tipus de classificador textual emprats, on es veu clarament que el Multinomial presenta uns millors resultats que el classificador de Bernouilli.



Gràfic 1. Comparativa de la precisió segons el tipus de classificador

A nivell conceptual, podem afirmar que a l'hora de predir, és més important la freqüència relativa amb la qual apareixen les diferents paraules a cadascuna de les notícies que les penalitzacions que s'assignen a cada notícia corresponents a la no presència de certes paraules. A priori no es pot suposar que un dels dos models sigui millor que l'altre, sinó que en la pròpia documentació[7] recomanen avaluar-los ambdós, ja que depenent del tipus de dades textuals i de la llargada dels texts, s'obtenen resultats de diferents magnituds. Tot i així, malgrat la conclusió sigui que és preferible escollir el classificador Multinomial per a l'anàlisi, les diferències no són prou elevades com per desaconsellar l'ús d'un classificador de Bernouilli. Com a norma general, el classificador de Bernouilli és millor per a texts curts, i a mesura que n'augmenta la llargària, l'efectivitat del Multinomial és més elevada (McCallum & Nigam, 1998).

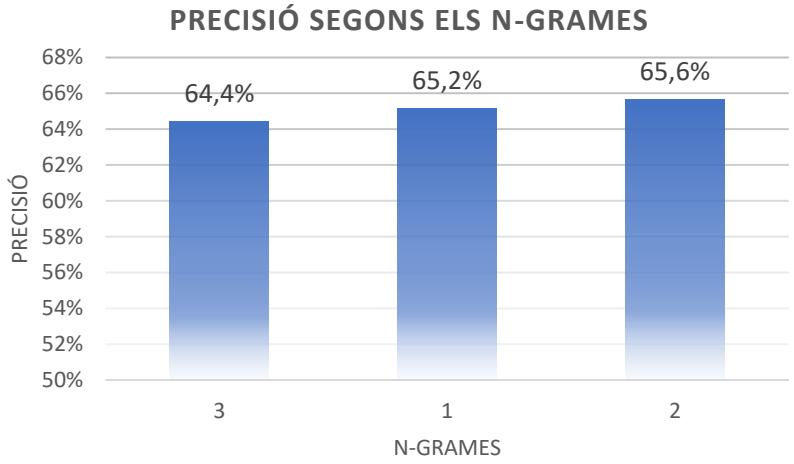
[5] (3 N-grames) * (8 Mínims Suports) * (9 Suavitzats) * (5 Mides Mostrals) * 15 Repeticions [Bernouilli] + (3 N-grames) * (8 Mínims Suports) * (5 Mides Mostrals) * 15 Repeticions [Multinomial] =18.000 Classificadors

[6] Intel Core i7-4790K - 4.00 GHz amb 32 GB RAM

[7] Es pot consultar a http://scikit-learn.org/stable/modules/naive_bayes.html

NOMBRE DE N-GRAMES

En aquest apartat s'estudia si les unitats (o variables) estudiades, és a dir, les x_i, permeten millorar el resultat predictiu al considerar agrupacions de dues o tres paraules com a unitats[8]. Això implica considerar parelles i paraules com a unitats, o triplets, parelles i paraules, respectivament.



Gràfic 2. Comparativa de la precisió segons el nombre de N-grames

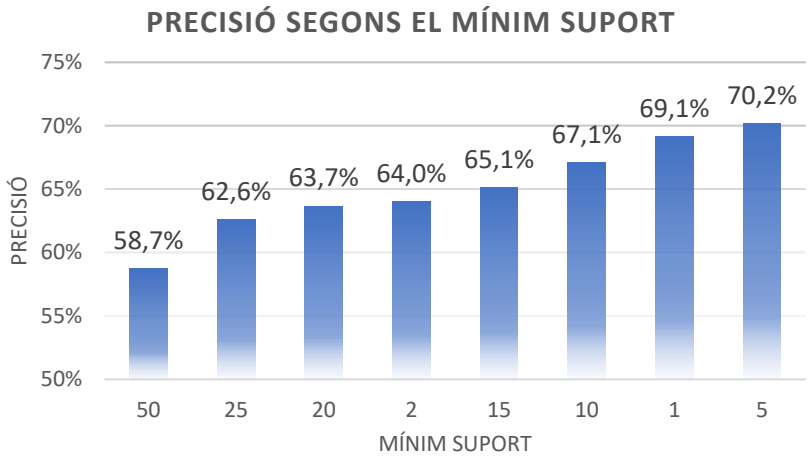
Malgrat la intuïció pugui portar a pensar que com més informació s'inclogui en les unitats explicatives, més informació puguem extreure de l'estil d'escriptura del diari, els resultats no són gens concloents en aquest aspecte, ja que les diferències entre el nombre màxim de n-grames considerats són molt petites, i no mostren una tendència ordenada. Donat que computacionalment és molt més costós ajustar un classificador per a un 3-grama que per a un 1-grama, si el temps és un factor important o la mida mostral és molt elevada (com és desitjable), la diferència de precisió no justifica la utilització d'un 3-grama.

NOMBRE MÍNIM DE SUPORT

L'avaluació del mínim suport no és més que el nombre mínim de vegades que ha d'aparèixer una paraula o grup de paraules a un diari concret per a ser considerat. És una mesura que permet obviar les paraules menys utilitzades i quedar-se amb les més representatives.

[8] Per exemple, una frase de 5 paraules diferents, que lògicament consta de 5 variables per a un classificador 1-grama, consta de 9 variables en el cas d'utilitzar fins a 2-grames, i de 12 en el cas de 3-grames. Un 3-grama de "El python és un programa", donaria les següents variables: "El", "python", "és", "un", "programa", "El python", "python és", "és un", "un programa", "El python és", "python és un", "és un programa".
En general, el nombre de variables a considerar amb un k-grama, si es disposa de n paraules és el següent:

$$n + (n-1) + \cdots + (n-k+1) = (n + (n-1) + \cdots + 1) - \big((n-k) + (n-k-1) + \cdots + 1\big) = \frac{n(n+1)-(n-k)(n-k+1)}{2}$$

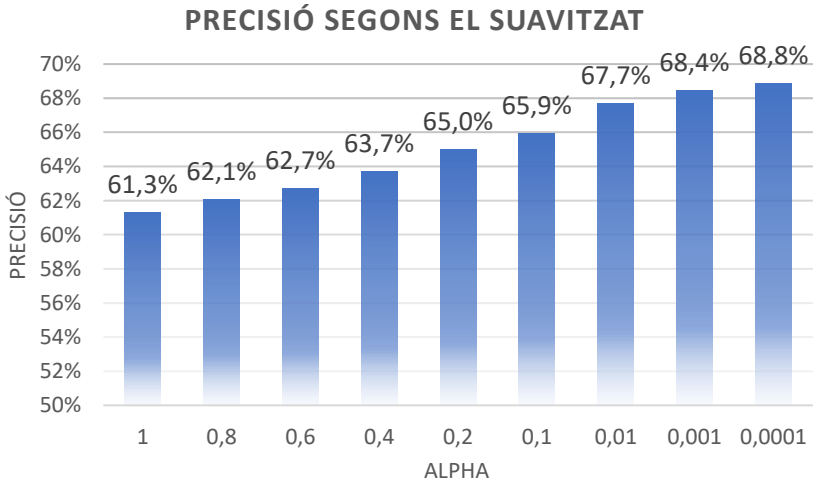


Gràfic 3. Comparativa de la precisió segons el suport de les variables

Els resultats, tot i així, no van a favor d'aplicar aquest tipus de filtre, ans al contrari. Si bé la tendència no és perfectament decreixent, sí que es pot afirmar que és preferible considerar el major nombre de paraules possible (i per tant un suport minin més petit), ja que ofereix millors resultats de precisió. Això implica que paraules que apareixen molt poc a un diari concret, poden ajudar a classificar molt millor una notícia que només emprar aquelles paraules que apareixen més freqüentment.

SUAVITZAT

Pels classificadors basats en el Naive Bayes Multinomial, s'han estudiat diversos paràmetres de suavitzat, posant especialment l'èmfasi en els valors propers al zero, ja que en l'estudi exploratori del paràmetre, van oferir millors resultats.



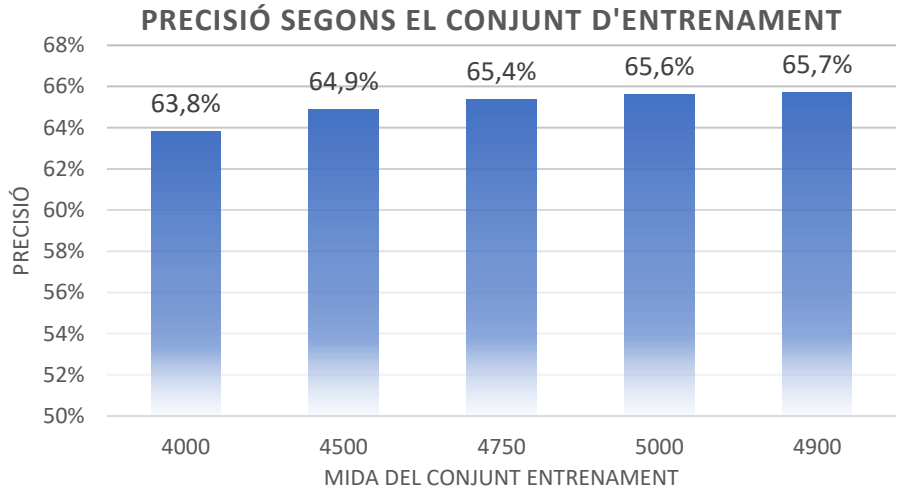
Gràfic 4. Comparativa de la precisió segons el paràmetre de suavitzat

Els resultats pel que fa al suavitzat són els més clars de tot el treball. Un α més proper a zero ofereix uns millors resultats de predicció en mitjana. Això implica que el potencial predictiu del classificador és més elevat quan (sempre que α > 0):

$$P(C_k)\prod_{i=1}^{n}\frac{N_{ki}+\alpha}{N_k+\alpha n}\xrightarrow{\alpha\to 0}P(C_k)\prod_{i=1}^{n}\frac{N_{ki}}{N_k}$$

MIDA DEL CONJUNT D'ENTRENAMENT

L'objectiu original d'aquest apartat era avaluar si hi ha una mida mínima de casos a seleccionar per al conjunt d'entrenament per a tal de garantir un bon funcionament del classificador. Una avaluació exploratòria d'aquesta hipòtesi va mostrar que conjunts d'entrenament relativament reduïts (menors de 1000 observacions), no presentaven una precisió superior al 50 %. Així doncs, buscant la precisió màxima, s'ha estudiat si aconseguir una major mostra, implica necessàriament obtenir uns millors valors de classificació. Els resultats obtinguts, són favorables a aquesta hipòtesi, ja que al augmentar la mida del conjunt d'entrenament, millora els resultats de predicció, el qual justifica obtenir una major mostra. Els resultats, en format gràfic, són els següents:



Gràfic 5. Comparativa de la precisió segons la mida del conjunt d'entrenament

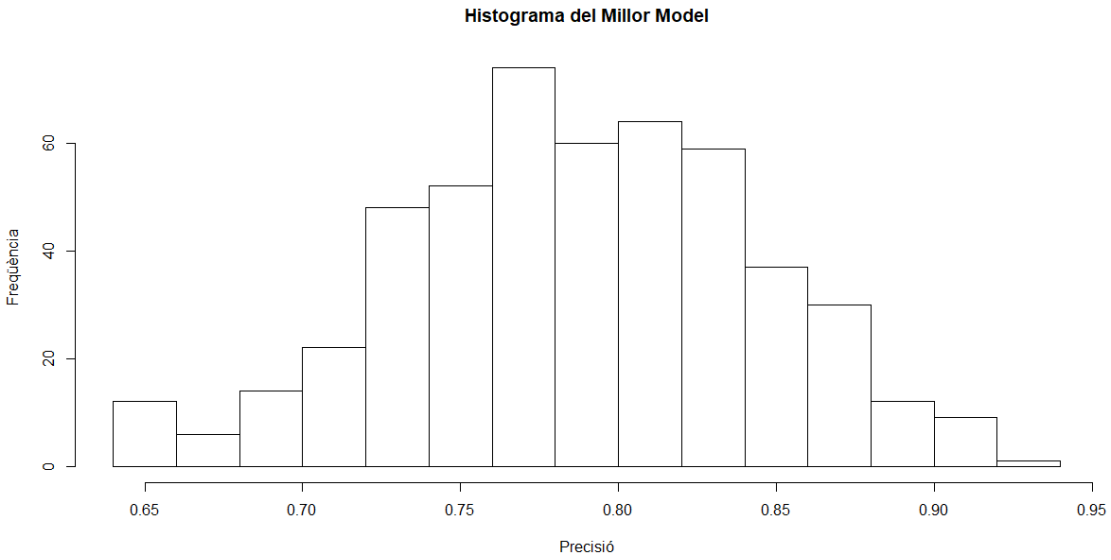
Si bé és cert que les diferències no són gaire acusades, hi ha una clara tendència creixent a millorar la precisió a major mostra, que es pot observar en la diferència d'incloure 500 notícies més al conjunt d'entrenament, el qual en l'entorn que estem tractant, augmenta en aproximadament un punt percentual al passar 4.000 a 4.500 notícies, i se suavitza lleugerament en el següent increment.

OPTIMITZACIÓ I MILLOR MODEL POSSIBLE

A continuació mostrem el millor classificador possible que s'ha obtingut en totes les simulacions realitzades, és a dir, aquell que ha predit en mitjana un major percentatge de notícies correctament de la mostra test.

Es tracta d'un classificador Multinomial amb $\alpha = 0.01$, entrenat 6.000 notícies, avaluant fins a 3-grames, i considerant totes les paraules independentment de la seva freqüència d'aparició (Suport mínim = 1)[9]. S'han realitzat 200 simulacions amb aquests paràmetres per a avaluar-ne l'estabilitat. La seva distribució pot veure's al gràfic 6. En el pitjor dels casos ha predit un 64 % de notícies correctament, i en el millor un 94 %, amb una mediana del 80 %.

[9] En memòria local ocupa 878.5 MB en format .pkl. L'opció de desar-lo permet reutilitzar-lo per a futures classificacions sense dependre de la mostra original.



Gràfic 6. Histograma de les Simulacions del Millor Model

Per a veure que la influència de la selecció del conjunt d'entrenament i el conjunt de test pot afectar més que la selecció concreta dels paràmetres, mostrem els 10 millors classificadors entrenats, segons la seva precisió mitjana:

Classificador	α	Mida Training	Suport	N-Grames	Precisió Mitjana
Multinomial	0,01	6000	1	3	**82,53**
Multinomial	0,001	6000	1	2	**81,33**
Multinomial	0,0001	6000	1	3	**81,12**
Multinomial	0,0001	6000	1	2	**80,93**
Bernouilli	-	6000	1	2	**80,66**
Multinomial	0,001	5900	1	2	**80,6**
Multinomial	0,01	5900	1	3	**80,53**
Multinomial	0,1	6000	1	2	**80,4**
Multinomial	0,1	5900	1	3	**80,4**
Multinomial	0,001	5750	1	3	**80,16**
Multinomial	0,001	5750	1	3	**80,13**

Taula 2. Millors Classificadors

Els millors classificadors entrenats, poden arribar a predir correctament els diaris amb una precisió superior al 80 %, que s'ha de comparar amb la probabilitat de classificar correctament un diari a l'atzar, que seria aproximadament 1/6 (16,67 %). Així doncs, es considera el resultat global de predicció molt satisfactori, ja que gairebé quintuplica el model nul.

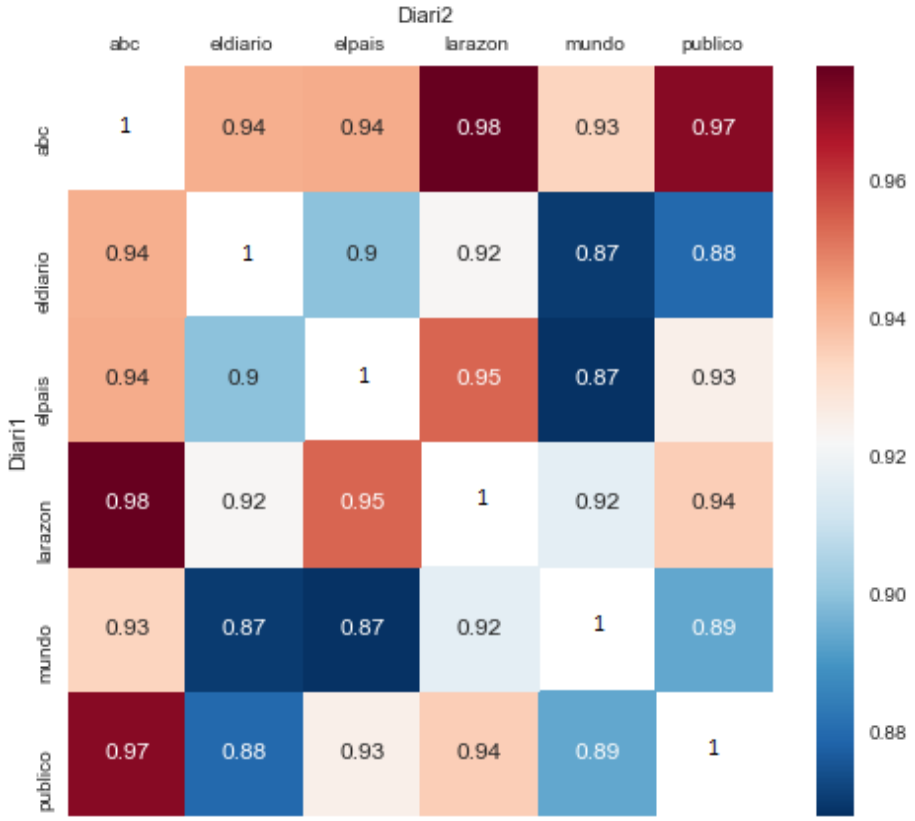
L'estudi de diferents paràmetres del model no és l'única manera de millorar-ne els resultats predictius. Hi ha autors (Weber et al., 2005) que proposen anivellar els pesos per a cadascuna de les classes estudiades, amb el qual es redueix l'error de classificació a la vegada que debilita l'assumpció d'independència entre variables. En aquest treball no s'ha realitzat aquesta transformació, ja que amb el model més simplificat ja s'obtenen resultats molt favorables, tot i que s'hauria de considerar com a possible millora dels models.

COMPARACIÓ DE DIARIS

En aquest apartat s'estudia l'eficiència del millor classificador trobat per a classificar entre parelles de diaris, o per a comparar entre l'estil d'escriptura d'un diari en relació al de tots els altres. L'objectiu és establir una mesura de proximitat entre diaris, és a dir, quins diaris utilitzen un estil d'escriptura més diferenciat entre ells. Per a validar aquests resultats, s'han creat 100 classificadors per a cada parell de diaris.

Individualment

Comparant les parelles de diaris individualment trobem que la precisió mitjana es dispara fins al 92 % (en blanc al gràfic 7). En primer lloc es pot afirmar que el classificador que s'ha construït per a comparacions múltiples, també dóna bons resultats per a classificar binàriament, i els millora.



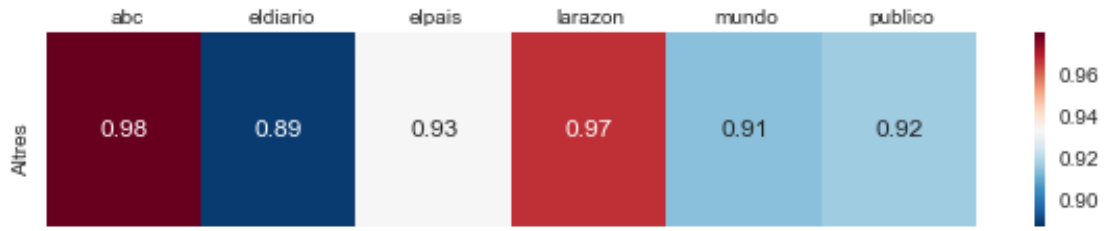
Gràfic 7. Comparació 1 vs 1

Per a avaluar quin diari té un estil d'escriptura més particular, s'ha proposat com a indicador la suma de les precisions en relació a tots els altres diaris. Així doncs, un valor molt elevat implica que el diari és fàcil de classificar correctament, i que per tant, té un estil d'escriptura més diferenciat. Utilitzant aquest criteri, de més diferent a menys trobem:

1. ABC
2. La Razón
3. Público
4. El País
5. El Diario
6. El Mundo

Agrupats

Per a avaluar l'estabilitat del resultat anterior, s'ha proposat un criteri alternatiu, que consisteix en comparar les notícies d'un diari, amb totes les notícies dels altres diaris, indistintament. Seria coherent que es mantingués un ordre similar a l'apartat anterior, com veiem a continuació.



Gràfic 8. Comparació 1 vs Tots

Com pot comprovar-se aquí, el nou ordre és el següent, de més fàcilment classificable a menys:

1. ABC
2. La Razón
3. El País
4. Público
5. El Mundo
6. El Diario

Així doncs, podem afirmar que els diaris amb un estil d'escriptura més particular, és a dir, més fàcilment classificable, són l'ABC i La Razón, mentre que els més difícilment identificables són El Mundo i El Diario.

RESULTATS FINALS

El resultat més evident del treball és que els classificadors basats en Naive Bayes són una molt bona eina per a classificar notícies basant-se només en les paraules utilitzades. Una de les seves principals virtuts és la senzillesa amb la qual es construeix, i que malgrat assumir unes condicions que no són certes, ofereix una precisió molt alta i uns resultats globals molt satisfactoris.

Observant els resultats de l'estudi marginal dels paràmetres pels classificadors i els millors classificadors entrenats, es poden donar algunes directrius per a obtenir uns millors resultats de classificació textual:

- Els classificadors Multinomials ofereixen millors resultats en mitjana que els classificadors Bernouilli, el qual implica que la freqüència relativa d'aparició de les paraules és més important que la no aparició de certes paraules en una notícia.
- A major mostra emprada en l'entrenament, millors resultats de classificació, el qual justifica invertir una major quantitat de temps en la recol·lecció de notícies.
- És preferible incloure totes les paraules independentment de la seva freqüència d'aparició.
- Considerar 2-grames o 3-grames dóna millors resultats puntuals màxims, però en mitjana no justifica el seu ús, donat que consumeix molta més memòria i temps, i no garanteix uns millors resultats.
- Un paràmetre de suavitzar proper a zero pels models Multinomials ofereix uns millors resultats que valors més elevats.

Considerant tots aquests factors, és possible entrenar un classificador que ofereixi un resultats encertats superiors al 80 % en mitjana, per a una mostra amb 6 categories diferents. Si es comparen parelles de diaris entre elles, la precisió pot arribar a superar el 95 % en diversos casos, el qual implica que es tracta d'una molt bona eina en termes de predicció. En referència als diaris concrets, aquells amb un estil d'escriptura més diferenciat dels demés són l'ABC i La Razón, i els que en tenen un menys definit són El Diario i El Mundo.

REFERÈNCIES

Hand, D., & Yu, K. (2001). Idiot's Bayes: Not So Stupid after All? *International Statistical Review / Revue Internationale De Statistique, 69*(3), 385-398.

McCallum A., & Nigam K. (1998). *A comparison of event models for Naive Bayes text classification*.

Metsis, V., et al. (2006). Spam Filtering with Naive Bayes - Which Naive Bayes? In *THIRD CONFERENCE ON EMAIL AND ANTI-SPAM (CEAS).*

Webb, G., Boughton, J. & Wang, Z. Mach Learn (2005). *Not So Naive Bayes: Aggregating One-Dependence Estimators*. 58: 5.

Enllaços

Documentació de la Biblioteca Beautiful Soup.

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Documentació de la Biblioteca Scikit-Learn.

http://scikit-learn.org/stable/modules/naive_bayes.html

Naive Bayes text classification. Universitat de Stanford.

https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html

Web Scraping. Wikipedia.

https://es.wikipedia.org/wiki/Web_scraping

ANNEXOS

Metadata

Títol: WEB SCRAPING AMB PYTHON. Una anàlisi textual de la premsa escrita

Autor: Joan Gasull Jolis

Tutor: Albert Ruiz Cirera

Resum: L'objectiu fonamental d'aquest treball és crear una eina que, donada una notícia, predigui a quin diari pertany. S'ha utilitzat "web scraping" per a obtenir una base de dades de notícies de forma massiva i el més automàtica possible. El classificador de notícies utilitza models basats en el Teorema de Bayes, i en el treball s'estudia com maximitzar-ne el potencial predictiu. S'arriba a la conclusió que, en els casos estudiats i només considerant el vocabulari utilitzat, aquesta tècnica de classificació dóna uns resultats molt precisos.

Resumen: El objetivo fundamental de este trabajo es crear una herramienta que, dada una noticia, prediga a qué diario pertenece. Se ha usado "web scraping" para obtener una base de datos de noticias de forma masiva y lo más automática posible. El clasificador de noticias usa modelos basados en el Teorema de Bayes, y en el trabajo se estudia como maximizar su potencial predictivo. Se llega a la conclusión que, en los casos estudiados y solo considerando el vocabulario usado, esta técnica de clasificación da unos resultados muy precisos.

Abstract: The main objective of this work is to create a tool that, given a news item, predicts which journal it belongs to. "Web scraping" has been used to get a news database in a massive way, as much automatic as possible. The news classifier uses models based on the Bayes' Theorem, and the paper studies how to maximize its predictive potential. To conclude, in the cases studied and only considering the vocabulary used, this classification technique gives very precise results.

Paraules Clau: Web Scraping, Naive Bayes, Anàlisi Textual, Classificació

Codi Emprat

SCRAPINGS I BASE DE DADES

El Diario

```
import time
from bs4 import BeautifulSoup
import urllib
import pandas as pd
from urlparse import urljoin
import requests
import tqdm

MAX=1000 #Nombre de noticies a recollir
s=set() #Conjunt on guardar els enllaços
i=1 #Iterador
while(len(s)<MAX):
   base_url = "http://www.eldiario.es/politica/?page=" + str(i) #Pagina on buscar els links
   response = requests.get(base_url)
   soup = BeautifulSoup(response.content, "html.parser")
   i=i+1
   for link in soup.findAll('a'): #Buscar els enllaços
      try: #En cas d'error no trenca el bucle
         url = link["href"]
         absolute_url = urljoin(base_url, url)
         if(absolute_url.startswith("http://www.eldiario.es/politica/") and not
absolute_url.startswith("http://www.eldiario.es/politica/?page=")):
            s.add(absolute_url) #Guardar els enllaços
      except:
         break;
   print(len(s))
s.remove("http://www.eldiario.es/politica/")
dataframeExist=False
for link in tqdm.tqdm(s): #Començar a llegir els enllaços per obtenir el text
   r = urllib.urlopen(link).read()
   soup = BeautifulSoup(r,'lxml')

   parr = soup.find_all('p', class_="mce") #Troba tot el text de la noticia
   noticia=""
   for m in parr: #Afegeix tots els paràgrafs a una sola variable
      noticia=noticia+str(m)
   noticia=noticia.decode('utf8') #Converteix a utf8 el text

   while(noticia.find("<")!=-1): #Elimina el codi html
      noticia=noticia[:noticia.find("<")]+noticia[noticia.find(">")+1:]

#Guarda les dades
   mydata = [{'NOTICIA': noticia},]
   if(dataframeExist==False):
      result = pd.DataFrame(mydata)
      dataframeExist=True
   else:
      df=pd.DataFrame(mydata)
      result = result.append(df,ignore_index=True)

result.to_csv("BBDDeldiario.csv",encoding="utf8")
```

ABC

```
from bs4 import BeautifulSoup
import urllib
import pandas as pd
from urlparse import urljoin
import requests
import tqdm

MAX=1000
s=set()
i=0
```

```
while(len(s)<MAX):
   base_url = “http://www.abc.es/politica/pag-” + str(i) + “.html”
   response = requests.get(base_url)
   soup = BeautifulSoup(response.content, “html.parser”)
   for link in soup.findAll('a'):
      try:
         url = link[“href”]
         absolute_url = urljoin(base_url, url)
         if(absolute_url.startswith(“http://www.abc.es/espana/”) or absolute_url.startswith(“http://www.abc.es/internacional/”)):
            s.add(absolute_url)
      except:
         break;
   base_url = “http://www.abc.es/internacional/pagina-” + str(i) + “.html”
   response = requests.get(base_url)
   soup = BeautifulSoup(response.content, “html.parser”)
   for link in soup.findAll('a'):
      try:
         url = link[“href”]
         absolute_url = urljoin(base_url, url)
         if(absolute_url.startswith(“http://www.abc.es/espana/”) or absolute_url.startswith(“http://www.abc.es/internacional/”)
and (url.find(“2015”)!=-1 or url.find(“2016”)!=-1 or url.find(“2017”)!=-1))):
            s.add(absolute_url)
      except:
         break;
   i=i+1
   print len(s)


dataframeExist=False
for link in tqdm.tqdm(s):
   r = urllib.urlopen(link).read()
   soup = BeautifulSoup(r,'lxml')

   parr = soup.find_all(class_=“col-A cuerpo-articulo gris-ultra-oscuro”)

   noticia=““
   for m in parr:
      noticia=noticia+str(m)
   noticia=noticia[noticia.find(“<p>“):noticia.find('</p></div><aside class=“temas-fin-cuerpo clear”>')]
   noticia=noticia+”> “
   while(noticia.find(“<“)!=-1):
         noticia=noticia[:noticia.find(“<“)]+noticia[noticia.find(“>“)+1:]
   noticia=noticia.replace(“>“,””)
   mydata = [{'NOTICIA': noticia},]
   if(dataframeExist==False):
      result = pd.DataFrame(mydata)
      dataframeExist=True
   else:
      df=pd.DataFrame(mydata)
      result = result.append(df,ignore_index=True)

result.to_csv(“BBDDabc.csv”,encoding=“utf8”)
```

Público

```
import time
from bs4 import BeautifulSoup
import urllib
import pandas as pd
from urlparse import urljoin
import requests
import tqdm

MAX=1000
s=set()
i=1
while(len(s)<MAX):
   base_url =“http://www.publico.es/page/ultimas-noticias-politica.html?page=“ + str(i)
   response = requests.get(base_url)
   soup = BeautifulSoup(response.content, “html.parser”)
   for link in soup.findAll('a'):
```

```
        try:
            url = link["href"]
            absolute_url = urljoin(base_url, url)
            if(absolute_url.startswith("http://www.publico.es/politica/") and len(s)<MAX):
                s.add(absolute_url)
        except:
            break;
    base_url ="http://www.publico.es/page/ultimas-noticias-internacional.html?page=" + str(i)
    response = requests.get(base_url)
    soup = BeautifulSoup(response.content, "html.parser")
    for link in soup.findAll('a'):
        try:
            url = link["href"]
            absolute_url = urljoin(base_url, url)
            if(absolute_url.startswith("http://www.publico.es/politica/") and len(s)<MAX):
                s.add(absolute_url)
        except:
            break;
    i=i+1
    print len(s)

dataframeExist=False

for link in tqdm.tqdm(s):
    r = urllib.urlopen(link).read()
    soup = BeautifulSoup(r,'lxml')
    parr = soup.find_all(class_="article-text")
    noticia=""
    for m in parr:
        noticia=noticia+str(m)
    while(noticia.find("<")!=-1):
        noticia=noticia[:noticia.find("<")]+noticia[noticia.find(">")+1:]
        mydata = [{'NOTICIA': noticia},]
    if(dataframeExist==False):
        result = pd.DataFrame(mydata)
        dataframeExist=True
    else:
        df=pd.DataFrame(mydata)
        result = result.append(df,ignore_index=True)
result.to_csv("BBDDpublico.csv",encoding="utf8")
```

El País

```
from bs4 import BeautifulSoup
import urllib
import pandas as pd
import tqdm
import time

LIMITLINKS=1000 #Nombre de links que volem obtenir
s=set() #Conjunt on es guarden els links
linkstoscrap=set(["http://politica.elpais.com/","http://internacional.elpais.com/","http://elpais.com/tag/fecha/hoy"])  #Afegim
seccions que ens interesen
#Links on començar

validlinks=["http://internacional.elpais.com/i",
"http://ccaa.elpais.com/c","http://politica.elpais.com/p","http://elpais.com/politica/2","http://elpais.com/internacional/2"]
#Començaments de links que son valids
filtros=["politica","internacional","ccaa"]
aux=0
url="http://politica.elpais.com/"
linksscrapped=set()
while(len(s)<LIMITLINKS): #Creacio del bucle que busca fins al màxim de links que volem obtenir
    try:
        r = urllib.urlopen(url).read()
        if(url not in linksscrapped):
            linksscrapped.add(url)
        soup = BeautifulSoup(r,'lxml')
        soup=str(soup)
        while(aux!=len(soup)): #El bucle busca enllaços fins que no en trobi cap mes
            aux=len(soup)
            soup=soup[soup.find('http'):]
```

```
        word=soup[:soup.find('"')]
        soup=soup[soup.find('"'):]
        if(word.find("pais") !=-1 and len(word)<300 and word.find("bloque_comentarios")==-1 and word.find("?")==-1 and
word.find("widget")==-1 and word.find(".amp.")==-1 and word.find("iconos")==-1 and word.find("plus")==-1 and
(url.find("2015")!=-1 or url.find("2016")!=-1 or url.find("2017")!=-1)): #Comprovacions de si l'enllaç es valid
            for validlink in validlinks:
                if(word.startswith(validlink)):
                    s.add(word) #Guarda l'enllaç d'interes
            if((word not in linkstoscrap) and (word not in linksscrapped)):
                for fil in filtros:
                    if(word.find(fil)!=-1):
                        linkstoscrap.add(word) #Guarda l'enllaç per buscar-hi enllaços
        url=linkstoscrap.pop() #Treu l'enllaç de la llista d'enllaços pendents
        print len(s)
    except:
        print "Esperant"
        url=linkstoscrap.pop() #Treu l'enllaç de la llista d'enllaços pendents

dataframeExist=False

for link in tqdm.tqdm(s):
    time.sleep(.2)
    try: #Prova d'exercutar el procés, en cas d'error no para el bucle
        r = urllib.urlopen(link).read()

        soup = BeautifulSoup(r,'lxml')
        parr = soup.find_all(class_="articulo-cuerpo" #Obte tot el text
        noticia=""
        for m in parr: #Ajunta els paragrafs
            noticia=noticia+str(m)
        while(noticia.find("<")!=-1): #Neteja el codi html
            noticia=noticia[:noticia.find("<")]+noticia[noticia.find(">")+1:]
        while(noticia.find("{")!=-1):
            noticia=noticia[:noticia.find("{")]+noticia[noticia.find("}")+1:]
        noticia.replace("MÁS INFORMACIÓN","") #Excepcio particular
       #Guarda les dades
        mydata = [{'NOTICIA': noticia},]
        if(dataframeExist==False):
            result = pd.DataFrame(mydata)
            dataframeExist=True
        else:
            df=pd.DataFrame(mydata)
            result = result.append(df,ignore_index=True)
    except:
        print "Error general"

result.to_csv("BBDDelpais.csv",encoding="utf8")
```

El Mundo

```
from bs4 import BeautifulSoup
import urllib
import pandas as pd
import tqdm

LIMITLINKS=1000
s=set()
linkstoscrap=["http://www.elmundo.es/espana.html","http://www.elmundo.es/internacional.html"]
validlinks=["http://www.elmundo.es/espana", "http://www.elmundo.es/internacional" , "http://www.elmundo.es/madrid",
"http://www.elmundo.es/andalucia","http://www.elmundo.es/catalunya","http://www.elmundo.es/comunidad-
valenciana","http://www.elmundo.es/pais-vasco"]
aux=0
url=linkstoscrap[0]
linksscrapped=set([linkstoscrap[0]])

while(len(s)<LIMITLINKS):
    r = urllib.urlopen(url).read()
    soup = BeautifulSoup(r,'lxml')
    soup=str(soup)
    while(len(soup)>100):
        aux=len(soup)
        soup=soup[soup.find('http'):]
```

```
      word=soup[:soup.find('"')]
      soup=soup[soup.find('"'):]
      for validlink in validlinks:
         if(word.startswith(validlink) and word.find("ancla_comentarios") == -1 and len(s)<LIMITLINKS and and (word.find("2015")!=-
1 or word.find("2016")!=-1 or word.find("2017")!=-1))):
            s.add(word)
            if((word not in linkstoscrap) and (word not in linksscrapped)):
               linkstoscrap.append(word)
               linksscrapped.add(word)
   url=linkstoscrap.pop()
   print len(s)
print s

dataframeExist=False
for link in tqdm.tqdm(s):
   r = urllib.urlopen(link).read()
   soup = BeautifulSoup(r,'lxml')
   parr = soup.find_all(class_="row content cols-30-70")
   noticia=""
   for m in parr:
      noticia=noticia+str(m)
      noticia=noticia[noticia.find("</time>"):]
      try:
         noticia=noticia[:noticia.find("<aside")]
      except:
         break
   noticia=noticia+">"
   while(noticia.find("<")!=-1):
      noticia=noticia[:noticia.find("<")]+noticia[noticia.find(">")+1:]
   noticia=noticia.replace(">","")
   mydata = [{'NOTICIA': noticia},]
   if(len(noticia)<100):
      print "Noticia massa curta"
   elif(dataframeExist==False):
      result = pd.DataFrame(mydata)
      dataframeExist=True
   else:
      df=pd.DataFrame(mydata)
      result = result.append(df,ignore_index=True)

result.to_csv("BBDDmundo.csv",encoding="utf8")
```

La Razón

```
from bs4 import BeautifulSoup
import urllib
import pandas as pd
import tqdm

LIMITLINKS=1000
s=set()
linkstoscrap=["http://www.larazon.es/espana","http://www.larazon.es/internacional"]
validlinks=["http://www.larazon.es/espana","http://www.larazon.es/local","http://www.larazon.es/internacional"]
aux=0
url=linkstoscrap[0]
linksscrapped=set([linkstoscrap[0]])

while(len(s)<LIMITLINKS):
   r = urllib.urlopen(url).read()
   soup = BeautifulSoup(r,'lxml')
   soup=str(soup)
   while(len(soup)>100):
      aux=len(soup)
      soup=soup[soup.find('href="')+6:]
      word=soup[:soup.find('"')]
      word="http://www.larazon.es"+word
      word=word.replace(" ","")
      soup=soup[soup.find('"'):]
      for validlink in validlinks:
         if(word.startswith(validlink) and len(s)<LIMITLINKS and (word.find("2015")!=-1 or word.find("2016")!=-1 or
word.find("2017")!=-1))):
            s.add(word)
```

```
            print len(s)
            if((word not in linkstoscrap) and (word not in linksscrapped)):
                linkstoscrap.append(word)
                linksscrapped.add(word)
    url=linkstoscrap.pop()
dataframeExist=False
for link in tqdm.tqdm(s):
    r = urllib.urlopen(link).read()
    soup = BeautifulSoup(r,'lxml')
    parr = soup.find_all(class_="text")
    noticia=""
    for m in parr:
        noticia=noticia+str(m)
    noticia=noticia+">"
    while(noticia.find("<")!=-1):
        noticia=noticia[:noticia.find("<")]+noticia[noticia.find(">")+1:]
    noticia=noticia.replace(">","")
    mydata = [{'NOTICIA': noticia},]
    if(len(noticia)<100):
        print "Noticia massa curta"
    elif(dataframeExist==False):
        result = pd.DataFrame(mydata)
        dataframeExist=True
    else:
        df=pd.DataFrame(mydata)
        result = result.append(df,ignore_index=True)

result.to_csv("BBDDlarazon.csv",encoding="utf8")
```

Combinar Bases de Dades

```
import pandas as pd
import string
import tqdm as tqdm

#Llegeix tots els fitxers per separat
df=pd.read_csv("BBDDeldiario.csv",index_col=0)
df['DIARI'] = pd.Series("eldiario", index=df.index)
df2=pd.read_csv("BBDDelpais.csv",index_col=0)
df2['DIARI'] = pd.Series("elpais", index=df2.index)
df3=pd.read_csv("BBDDabc.csv",index_col=0)
df3['DIARI'] = pd.Series("abc", index=df3.index)
df4=pd.read_csv("BBDDpublico.csv",index_col=0)
df4['DIARI'] = pd.Series("publico", index=df4.index)
df5=pd.read_csv("BBDDmundo.csv",index_col=0)
df5['DIARI'] = pd.Series("mundo", index=df5.index)
df6=pd.read_csv("BBDDlarazon.csv",index_col=0)
df6['DIARI'] = pd.Series("larazon", index=df5.index)
df=df.append(df2,ignore_index=True)
df=df.append(df3,ignore_index=True)
df=df.append(df4,ignore_index=True)
df=df.append(df5,ignore_index=True)
df=df.append(df6,ignore_index=True)
df = df[df["NOTICIA"] != 0]

#Elimina els noticies que no s'ha emmagatzemat correctament
for i in tqdm.tqdm(range(0, len(df))):
    try:
        if(not(isinstance(df.iloc[i]['NOTICIA'], str))):
            df=df.drop(df.index[i])
            i=i-1
    except:
        break;
#Correcions particulars i elimina noticies amb massa poc text recollit
for i in tqdm.tqdm(range(0, len(df)+1000)):
    if(i>=len(df)): #S'assegura que el bucle no es surt dels limits
        break;
    if("articleBody" in df.iloc[i]['NOTICIA']):
        df=df.drop(df.index[i])
    if(len(df.iloc[i]['NOTICIA'])<100):
        df=df.drop(df.index[i])
```

```
#Elimina tabuladors, salts de línia, signes de puntuació i digits
for i in tqdm.tqdm(range(0, len(df))):
   df.iloc[i]['NOTICIA']=df.iloc[i]['NOTICIA'].replace("\t","")
   df.iloc[i]['NOTICIA']=df.iloc[i]['NOTICIA'].replace("\r","")
   df.iloc[i]['NOTICIA']=df.iloc[i]['NOTICIA'].replace("\n","")
   for word in string.punctuation:
      df.iloc[i]['NOTICIA']=df.iloc[i]['NOTICIA'].replace(word,"")
   for num in string.digits:
      df.iloc[i]['NOTICIA']=df.iloc[i]['NOTICIA'].replace(num,"")

df=df.dropna(subset = ['NOTICIA', 'DIARI']) #Elimina NAs
df=df.sample(frac=1) #Reordena la base de dades
df.to_csv("BBDDcompleta.csv",encoding="utf8") #Guarda la base de dades
#Calcula la llargada dels articles
variable=list()
for i in xrange(0,len(df)):
   variable.append(len(df.iloc[i]["NOTICIA"]))
df['LEN']=variable
print df.groupby(['DIARI'])['LEN'].mean()
```

CLASSIFICADORS

Simulació General

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
import random as rd
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
from sklearn import metrics

df=pd.read_csv("BBDDcompleta.csv",index_col=0)
df=df.dropna(subset = ['NOTICIA', 'DIARI']) #Elimina NAs
N=15 #Nombre de repeticions per cada configuració de paràmetres
nblist=[MultinomialNB(),BernoulliNB()] #Tipus de classificadors
ngramlist=list(range(1, 4, 1)) #N-grames a considerar
minlist=[50,1,2,5,10,15,20,25] #Minim suport
alphas=list(range(0, 11, 2)) #Suavitzats
alphas.remove(0.0)
for i in range(0,len(alphas)):
   alphas[i]=float(alphas[i])/10
alphas.append(0.01)
alphas.append(0.001)
alphas.append(0.0001)
alphas.append(0.1)
samplesizes=[50,100,250,500,1000]
dataframeExist=False

#Bucle general
for minim in minlist:
   for sample in samplesizes:
      for ngram in ngramlist:
         for classificador in nblist:
            for alpha in alphas:
               x=list()
               for i in range(0,N):
                  rows = rd.sample(df.index, sample)
                  classificador.alpha=alpha #No afecta si es Bernouilli
                  y = df["DIARI"]
                  raw_test = df.ix[rows] #Conjunt d'avaluacio
                  raw_train = df.drop(rows) #Conjunt d'entrenament
                  y_train = raw_train["DIARI"]
                  y_test = raw_test["DIARI"]
                  #Creem l'eina que secciona el text i l'emmagatzema a la matriu
                  vectorizer = CountVectorizer(min_df=minim,
                   stop_words=stopwords.words('spanish'),
                   strip_accents='unicode',
                   ngram_range=(1,ngram)
                   )
```

```
            X_train = vectorizer.fit_transform(raw_train["NOTICIA"]) #Apliquem l'eina al conjunt d'entrenament
            X_test = vectorizer.transform(raw_test["NOTICIA"]) #Apliquem l'eina al conjunt d'avaluacio
            nb = classificador #Creem el classificador
            nb.fit(X_train,y_train) #L'entrenem

            y_hat = nb.predict(X_test) #Mirem els resultats que prediu sobre el conjunt d'avaluacio
            x.append(metrics.accuracy_score(y_hat, y_test)) #Precisió segons el valor que li correspon realment i la guardem
        if(classificador == BernouilliNB()):
            break; #Si el classificador es de Bernouilli no ens cal iterar pel parametre de suavitzat
        #Guardem les  dades de cada combinació de parametres
        classificadorstr=str(classificador)[:str(classificador).find("(")]
        mydata=[{'Mida Mostral': sample,'Alpha':alpha, 'Minim Suport': minim, "Classificador": classificadorstr,
"Ngrames":ngram, 'Mean': np.mean(x),'Min': np.min(x),'Max': np.max(x),'Variance': np.var(x),'Median': np.median(x)}]
        if(dataframeExist==False):
            result = pd.DataFrame(mydata)
            dataframeExist=True
        else:
            dfsupport=pd.DataFrame(mydata)
            result = result.append(dfsupport,ignore_index=True)

result.to_csv("ResumClassificadors_FINAL_05_05.csv",encoding="utf8")
print result.sort(columns="Mean",ascending=False)
```

Classificador Final

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
import random as rd
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
from sklearn import metrics
import matplotlib.pyplot as plt
import pickle
import tqdm as tqdm

N=200 #Nombre d'iteracions que volem
x=list()
df=pd.read_csv("BBDDcompleta.csv",index_col=0)
df=df.dropna(subset = ['NOTICIA', 'DIARI'])
vectorizer = CountVectorizer(min_df=1,
    stop_words=stopwords.words('spanish'),
    strip_accents='unicode',
    ngram_range=(1,3)
    )

for i in tqdm.tqdm(xrange(0,N)):
   rows = rd.sample(df.index, 50)
   y = df["DIARI"]
   raw_test = df.ix[rows]
   raw_train = df.drop(rows)
   y_train = raw_train["DIARI"]
   y_test = raw_test["DIARI"]
   X_train = vectorizer.fit_transform(raw_train["NOTICIA"])
   X_test = vectorizer.transform(raw_test["NOTICIA"])
   nb = MultinomialNB(alpha=0.0001)
   nb.fit(X_train,y_train)
   y_hat = nb.predict(X_test)
   x.append(metrics.accuracy_score(y_hat, y_test))

mylist=x
f = open("histograma.txt", "w")
f.write("\n".join(map(lambda x: str(x), mylist)))
f.close()

with open('model.pkl', 'wb') as fout:
  pickle.dump((vectorizer, nb), fout)
```

Classificador 1vs1

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
import random as rd
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
from sklearn import metrics
import tqdm

#Paramentres que millor s'han comportat a la simulació general
rangeMAX=3
supportMAX=1
classificador=MultinomialNB(alpha=0.01)
dataframeExist=False
mostra=50
N=100 #Nombre de repeticions per a cada parella de diaris
df=pd.read_csv("BBDDcompleta.csv",index_col=0)
df=df.dropna(subset = ['NOTICIA', 'DIARI'])
noms=df["DIARI"].unique()
noms2=noms
noms2=list(noms2)
for nom in tqdm.tqdm(noms): #S'itera sobre els noms dels diaris sense fer la comparació amb un mateix
   noms2.remove(nom)
   for nom2 in noms2:
      x=list()
      for i in range(0,N):
         df2=df[df["DIARI"].isin([nom,nom2])] #Nomes considerem la BBDD amb els dos diaris seleccionats
         rows = rd.sample(df2.index, mostra)
         y = df2["DIARI"]
         raw_test = df2.ix[rows]
         raw_train = df2.drop(rows)

         y_train = raw_train["DIARI"]
         y_test = raw_test["DIARI"]
         vectorizer = CountVectorizer(min_df=supportMAX,
          stop_words=stopwords.words('spanish'),
          strip_accents='unicode',
          ngram_range=(1,rangeMAX)
          )
         X_train = vectorizer.fit_transform(raw_train["NOTICIA"])
         X_test = vectorizer.transform(raw_test["NOTICIA"])
         nb = classificador
         nb.fit(X_train,y_train)
         y_hat = nb.predict(X_test)
         x.append(metrics.accuracy_score(y_hat, y_test))
      mydata=[{'Classificador': str(classificador), 'Diari1': nom, 'Diari2': nom2,'Mean': np.mean(x),'Min': np.min(x),'Max':
np.max(x),'Variance': np.var(x)}]
      if(dataframeExist==False):
         result = pd.DataFrame(mydata)
         dataframeExist=True
      else:
         dfsupport=pd.DataFrame(mydata)
         result = result.append(dfsupport,ignore_index=True)
result.to_csv("1vs2.csv",encoding="utf8")
print result.sort(columns="Mean",ascending=False)
```

Classificador 1vsTots

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
import random as rd
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords
from sklearn import metrics
import tqdm

rangeMAX=3
```

```
supportMAX=1
classificador=MultinomialNB(alpha=0.01)
dataframeExist=False
mostra=50
N=100
df=pd.read_csv("BBDDcompleta.csv",index_col=0)
df=df.dropna(subset = ['NOTICIA', 'DIARI'])
noms=df["DIARI"].unique()
for nom in tqdm.tqdm(noms):
   x=list()
   df2=df.copy(deep=True)
   df2.loc[df2['DIARI'] != nom,'DIARI'] = 'altres' #Recategoritzem tots els diaris que no siguin el que volem comparar amb els altres
   for i in range(0,N):
      rows = rd.sample(df2.index, mostra)
      y = df2["DIARI"]
      raw_test = df2.ix[rows]
      raw_train = df2.drop(rows)
      y_train = raw_train["DIARI"]
      y_test = raw_test["DIARI"]
      vectorizer = CountVectorizer(min_df=supportMAX,
      stop_words=stopwords.words('spanish'),
      strip_accents='unicode',
      ngram_range=(1,rangeMAX)
      )
      X_train = vectorizer.fit_transform(raw_train["NOTICIA"])
      X_test = vectorizer.transform(raw_test["NOTICIA"])

      nb = classificador
      nb.fit(X_train,y_train)
      y_hat = nb.predict(X_test)
      x.append(metrics.accuracy_score(y_hat, y_test))
   mydata=[{'Classificador': str(classificador), 'Diari1': nom,'Mean': np.mean(x),'Min': np.min(x),'Max': np.max(x),'Variance':
np.var(x)}]
   if(dataframeExist==False):
      result = pd.DataFrame(mydata)
      dataframeExist=True
   else:
      dfsupport=pd.DataFrame(mydata)
      result = result.append(dfsupport,ignore_index=True)

result.to_csv("1vsALL.csv",encoding="utf8")
print result.sort(columns="Mean",ascending=False)
```

RESULTATS

Resum dels Resultats

```
import pandas as pd
import numpy as np
from scipy import stats

df=pd.read_csv("BBDDRESULTATS2.csv",index_col=0,sep=";")
#Generem els resultats per a cada paramtre d'interes
classificador=df.groupby(['Classificador'])['Mean']
alpha=df.groupby(['Alpha'])['Mean']
minsuport=df.groupby(['Minim Suport'])['Mean']
ngrames=df.groupby(['Ngrames'])['Mean']
mostra=df.groupby(['Mida Mostral'])['Mean']
classificador.mean()
alpha.mean()
minsuport.mean()
ngrames.mean()
mostra.mean()
#Obtenim el millor classificador possible
print df.ix[df["Mean"].idxmax()]
```

Mapes de Calor

```
import seaborn as sns
import pandas as pd
import numpy as np
import pylab as pylab
```

```
import matplotlib.pyplot as plt

sns.set_palette("deep", desat=.6)
sns.set_context(rc={"figure.figsize": (8, 7)})
sns.set_style("whitegrid")
df=pd.read_csv("1vs2.csv",index_col=0)
for name in df.Diari1.unique():
    if name not in df.Diari2.unique():
        name2=df["Diari2"][df["Diari1"]==name]
        newrow=df[df["Diari1"]==name]
        newrow["Diari1"]=name2
        newrow["Diari2"]=name
        df=df.append(newrow)
for name in df.Diari2.unique():
    if name not in df.Diari1.unique():
        name2=df["Diari1"][df["Diari2"]==name]
        newrow=df[df["Diari2"]==name]
        newrow["Diari2"]=name2
        newrow["Diari1"]=name
        df=df.append(newrow)

rect = df.pivot("Diari1", "Diari2", "Mean")
x=list()
for name in df.Diari2.unique():
    for name2 in df.Diari2.unique():
        if(name==name2):
            rect.loc[name,name]=np.nan
        elif(np.isnan(rect.loc[name,name2])):
            rect.loc[name,name2]=rect.loc[name2,name]
        x.append(rect.loc[name,name2])
x = [y for y in x if str(y) != 'nan']
mitjana=np.mean(x)
diaris=pd.DataFrame(columns=['Diari', 'Suma'])
for name in df.Diari2.unique():
    diaris=diaris.append({'Diari': name, 'Suma': np.sum(rect[name])},ignore_index=True)
print diaris.sort(columns="Suma",ascending=False)
plot1=sns.heatmap(rect,annot=True,center=mitjana)
pylab.savefig("plot1.png")
df2=pd.read_csv("1vsALL.csv",index_col=0)
df2["Altres"]="Altres"
rect2 = df2.pivot("Altres","Diari1","Mean")
sns.set_context(rc={"figure.figsize": (10, 1.7)})
plot2=sns.heatmap(rect2,annot=True,center=np.mean(df2["Mean"]))
pylab.savefig("plot2.png")
```